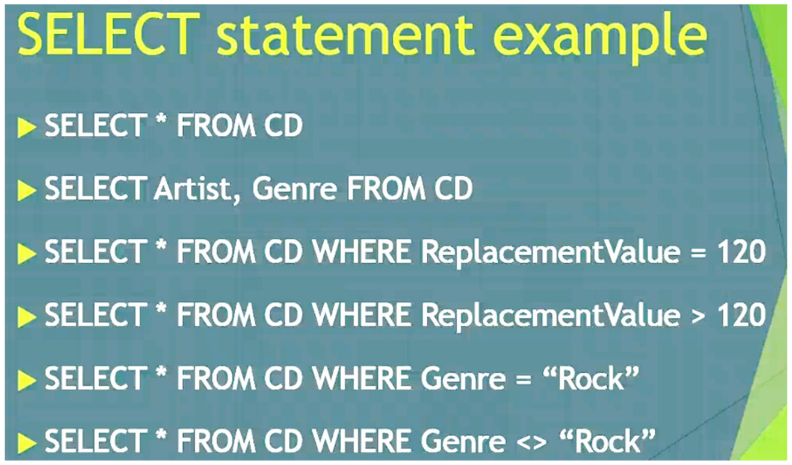
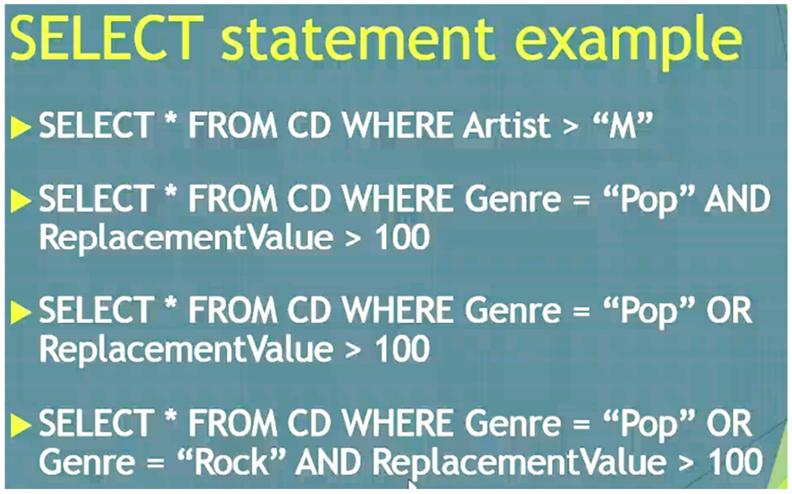
SQL Statements



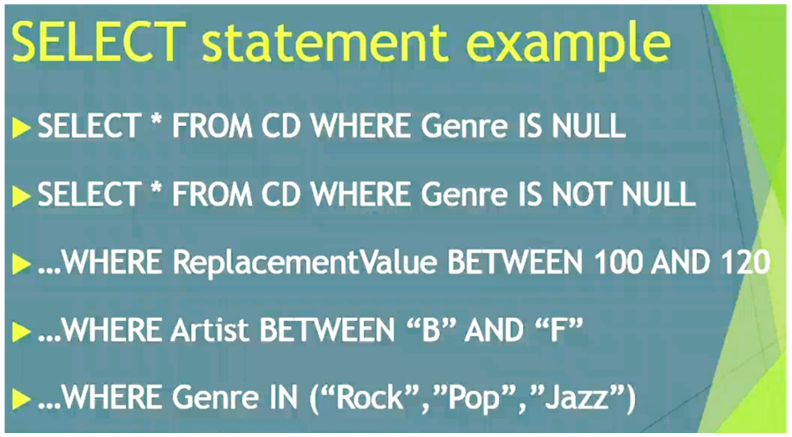
SELECT statement example

- SELECT * FROM CD
- SELECT Artist, Genre FROM CD
- SELECT * FROM CD WHERE ReplacementValue = 120
- SELECT * FROM CD WHERE ReplacementValue > 120
- SELECT * FROM CD WHERE Genre = "Rock"
- SELECT * FROM CD WHERE Genre <> "Rock"



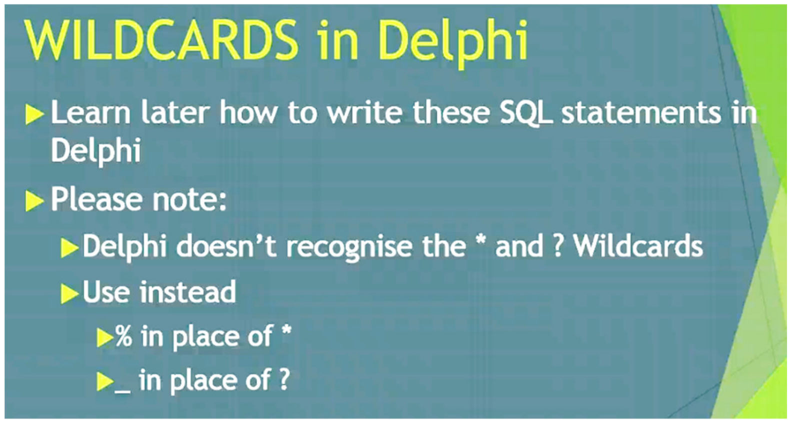
SELECT statement example

- SELECT * FROM CD WHERE Artist > "M"
- SELECT * FROM CD WHERE Genre = "Pop" AND ReplacementValue > 100
- SELECT * FROM CD WHERE Genre = "Pop" OR ReplacementValue > 100
- SELECT * FROM CD WHERE Genre = "Pop" OR Genre = "Rock" AND ReplacementValue > 100



SELECT statement example

- SELECT * FROM CD WHERE Genre IS NULL
- SELECT * FROM CD WHERE Genre IS NOT NULL
- ...WHERE ReplacementValue BETWEEN 100 AND 120
- ...WHERE Artist BETWEEN "B" AND "F"
- ...WHERE Genre IN ("Rock","Pop","Jazz")



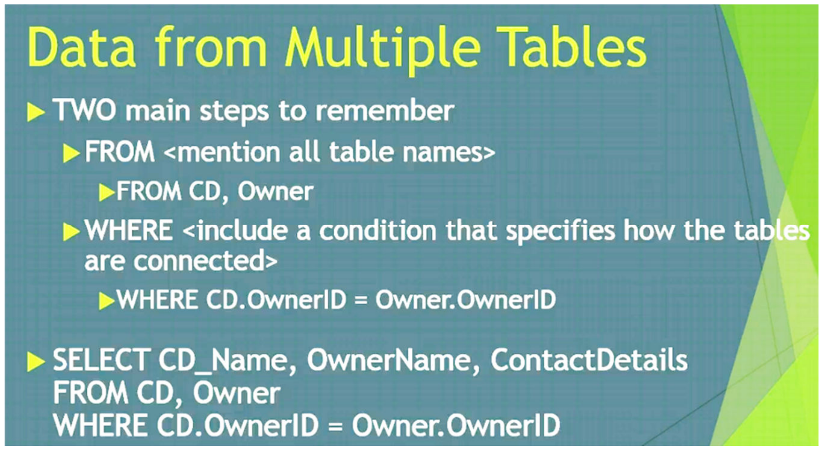
WILDCARDS in Delphi

- Learn later how to write these SQL statements in Delphi
- Please note:
  - Delphi doesn't recognise the * and ? Wildcards
  - Use instead
    - % in place of *
    - _ in place of ?

Examples:

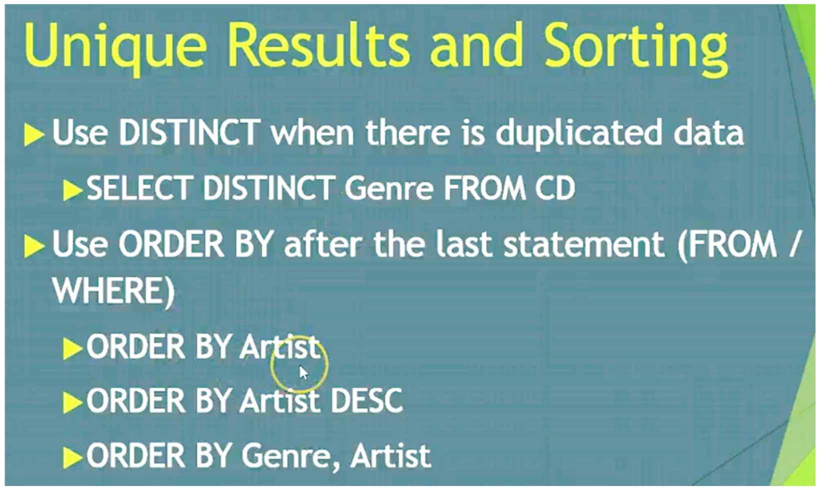
… WHERE Artist LIKE 'S%' - start with S
… WHERE Artist LIKE '%s' – ends with s
… WHERE Artist LIKE '%s%' – contains s anywhere in artist name

SQL Statements



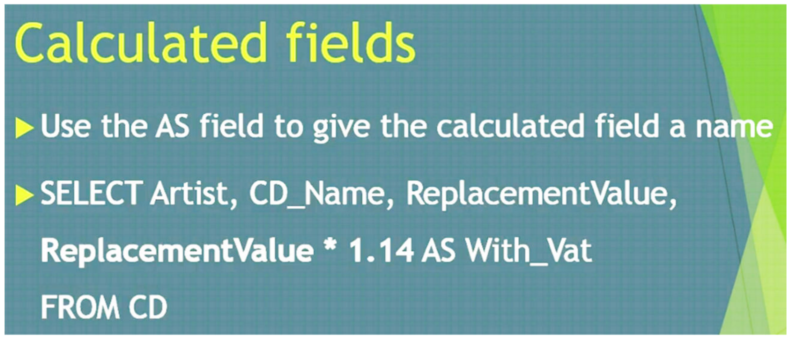
Data from Multiple Tables

- TWO main steps to remember
  - FROM <mention all table names>
    - FROM CD, Owner
  - WHERE <include a condition that specifies how the tables are connected>
    - WHERE CD.OwnerID = Owner.OwnerID
- SELECT CD_Name, OwnerName, ContactDetails
  FROM CD, Owner
  WHERE CD.OwnerID = Owner.OwnerID



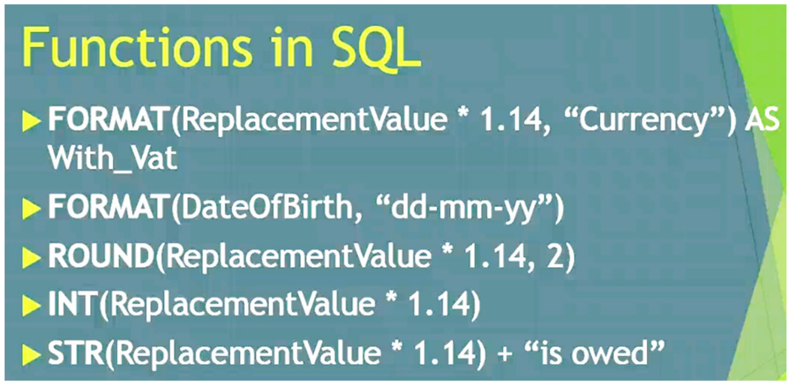
Unique Results and Sorting

- Use DISTINCT when there is duplicated data
  - SELECT DISTINCT Genre FROM CD
- Use ORDER BY after the last statement (FROM / WHERE)
  - ORDER BY Artist
  - ORDER BY Artist DESC
  - ORDER BY Genre, Artist



Calculated fields

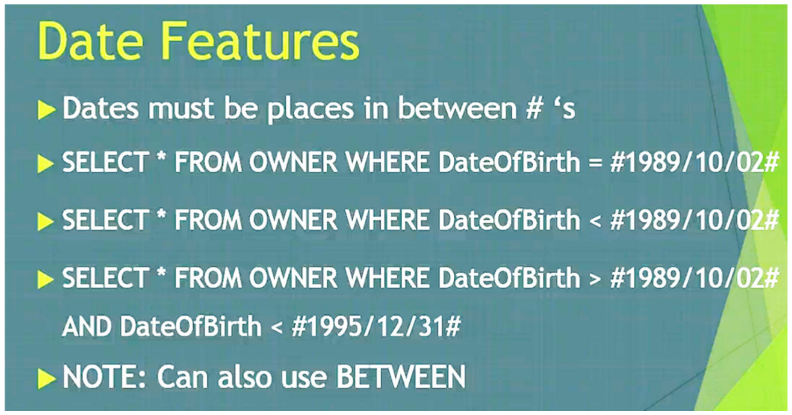
- Use the AS field to give the calculated field a name
- SELECT Artist, CD_Name, ReplacementValue,
  ReplacementValue * 1.14 AS With_Vat
  FROM CD



Functions in SQL

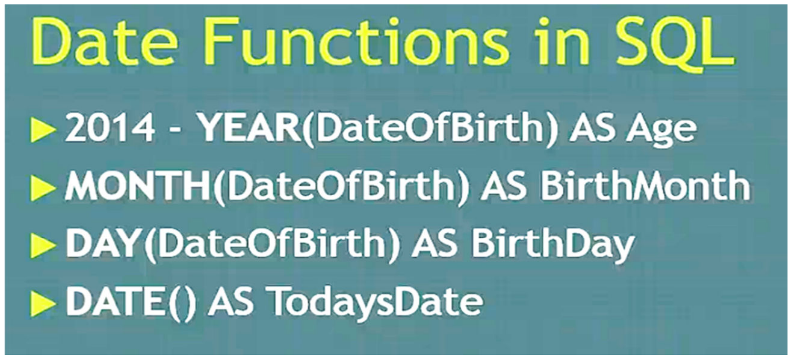
- **FORMAT**(ReplacementValue * 1.14, “Currency”) AS With_Vat
- **FORMAT**(DateOfBirth, “dd-mm-yy”)
- **ROUND**(ReplacementValue * 1.14, 2)
- **INT**(ReplacementValue * 1.14)
- **STR**(ReplacementValue * 1.14) + “is owed”

SQL Statements



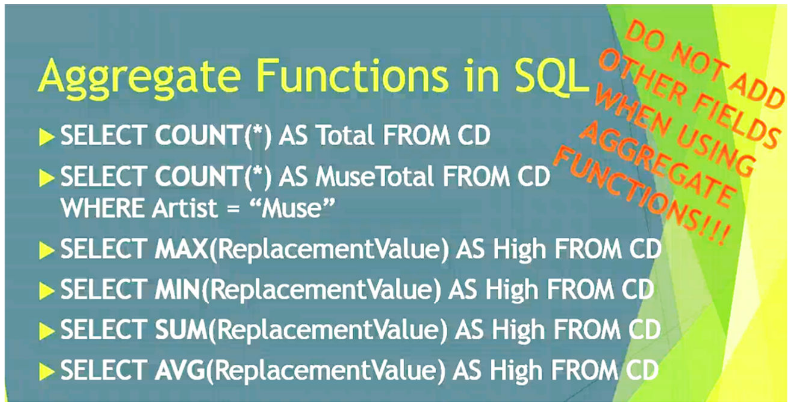
Date Features

- Dates must be places in between # ‘s
- SELECT * FROM OWNER WHERE DateOfBirth = #1989/10/02#
- SELECT * FROM OWNER WHERE DateOfBirth < #1989/10/02#
- SELECT * FROM OWNER WHERE DateOfBirth > #1989/10/02#
  AND DateOfBirth < #1995/12/31#
- NOTE: Can also use BETWEEN



Date Functions in SQL

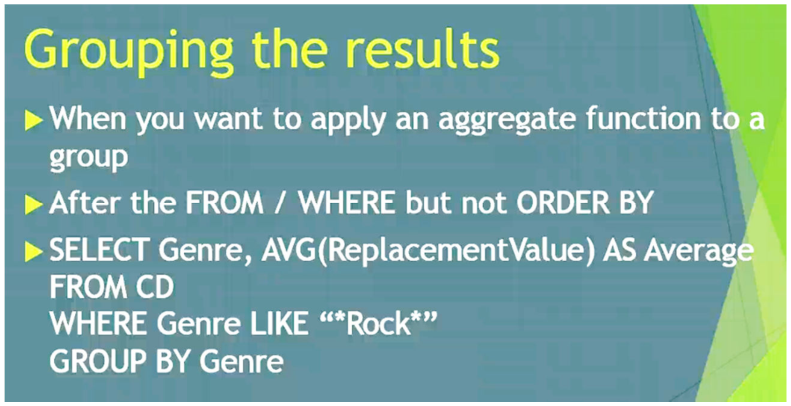
- 2014 - YEAR(DateOfBirth) AS Age
- MONTH(DateOfBirth) AS BirthMonth
- DAY(DateOfBirth) AS BirthDay
- DATE() AS TodaysDate



Aggregate Functions in SQL

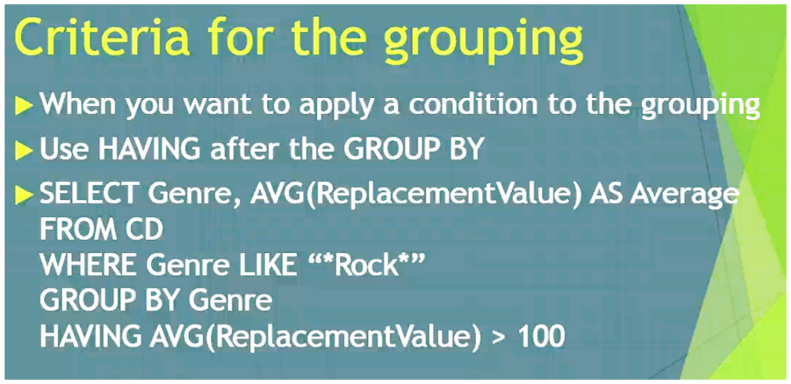
DO NOT ADD OTHER FIELDS WHEN USING AGGREGATE FUNCTIONS!!!

- SELECT COUNT(*) AS Total FROM CD
- SELECT COUNT(*) AS MuseTotal FROM CD
  WHERE Artist = “Muse”
- SELECT MAX(ReplacementValue) AS High FROM CD
- SELECT MIN(ReplacementValue) AS High FROM CD
- SELECT SUM(ReplacementValue) AS High FROM CD
- SELECT AVG(ReplacementValue) AS High FROM CD



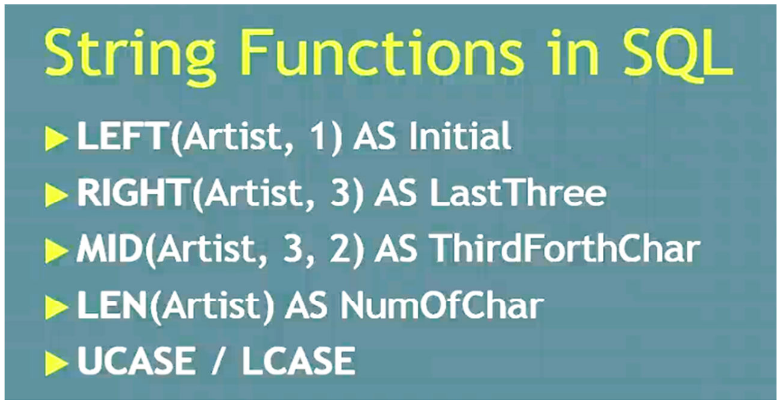
Grouping the results

- When you want to apply an aggregate function to a group
- After the FROM / WHERE but not ORDER BY
- SELECT Genre, AVG(ReplacementValue) AS Average
  FROM CD
  WHERE Genre LIKE “*Rock*”
  GROUP BY Genre



Criteria for the grouping

- When you want to apply a condition to the grouping
- Use HAVING after the GROUP BY
- SELECT Genre, AVG(ReplacementValue) AS Average
  FROM CD
  WHERE Genre LIKE “*Rock*”
  GROUP BY Genre
  HAVING AVG(ReplacementValue) > 100



String Functions in SQL

- LEFT(Artist, 1) AS Initial
- RIGHT(Artist, 3) AS LastThree
- MID(Artist, 3, 2) AS ThirdForthChar
- LEN(Artist) AS NumOfChar
- UCASE / LCASE